# Product Requirements Document (PRD): NearDrop

## Product Summary

NearDrop is a modern, privacy-focused communication app built for Apple devices. It enables encrypted, peer-to-peer messaging and file transfer via MultipeerConnectivity, similar to AirDrop but with end-to-end encryption and user identity verification.

## Objectives

1. Secure, serverless local communication.  
2. 1:1 and group messaging (2–8 peers).  
3. Encrypted file and text transfers.  
4. AirDrop-like UX with persistent encryption.  
5. Built entirely on Apple’s modern frameworks.

## Target Platforms & Stack

Language: Swift 5.10+  
UI: SwiftUI  
Networking: MultipeerConnectivity  
Encryption: CryptoKit (X25519, Ed25519, AES-GCM)  
Storage: Core Data / SQLite  
Persistence: Keychain + Secure Enclave  
Architecture: MVVM + Swift Concurrency  
Testing: XCTest

## Core Features

1. 1:1 encrypted messaging.  
2. Group chats up to 8 peers.  
3. Encrypted file sharing.  
4. Offline-first, serverless.  
5. Safety code verification.  
6. Ephemeral messaging (24h/7d).

## Extended Features (V1+)

• Crypto payload transfer (signed transactions).  
• Media & voice notes.  
• QR device linking.  
• Encrypted contact sharing.  
• AirDrop integration for .ndmsg files.

## Key Use Cases

• Offline chat  
• Secure credential handoff  
• Ad-hoc team comms  
• Encrypted crypto payload exchange

## Security Model

• Identity: Ed25519 signing  
• Session: X25519 ephemeral keys  
• Encryption: AES-GCM  
• Key derivation: HKDF-SHA256  
• Safety codes for verification  
• On-device Keychain storage

## UX & Interface

Clean, Apple-like UI with Discovery, Chat, Group Chat, and Settings screens.  
Minimalist SwiftUI design using system colors and typography.

## Data & Privacy

All data stored locally and encrypted.  
No servers or analytics by default.  
Optional opt-in telemetry (aggregated only).

## Developer Notes

• Use SwiftUI universal project.  
• Implement MPC peer discovery + crypto handshake.  
• Build 1:1 encrypted chat first.  
• Add Core Data persistence.  
• Use async/await.  
• Add XCTest for crypto + reliability.

## Roadmap

Phase 1: Encrypted 1:1 chat (4 weeks)  
Phase 2: Group chat + attachments (8 weeks)  
Phase 3: Crypto payloads (12 weeks)  
Phase 4: UI polish + App Store prep (14+ weeks)

## Risks & Mitigation

• MPC instability → retry logic  
• Key mismatch UX → simple safety codes  
• App Store crypto policy → position as messaging app  
• Battery → limit background discovery

## Deliverables for Claude Code

1. SwiftUI universal app scaffold.  
2. CryptoKitWrapper.swift  
3. PeerService.swift (MPC)  
4. MessageStore.swift  
5. ChatView.swift  
6. Working 1:1 encryption prototype.

## Acceptance Criteria

✅ Devices discover/connect  
✅ Safety codes match  
✅ Messages decrypt correctly  
✅ Attachments transfer securely  
✅ 100% offline  
✅ Privacy-compliant

## Future Opportunities

• Offline asset handoff  
• Local web3 broadcast  
• Apple Watch integration  
• Enterprise/MDM edition